Hermite

Sean Laverty

10/2/2019

The RMarkdown format could be used to effectively prepare your homework - this template is a reasonable starting point that you are free to use. You could define and print a function early on, then in separate uses apply a function to different problems. This isn't the best implementation - it isn't meant to be - but shows a variety of elements that can be combined to effectively prepare your work.

Some problems will require written work, but you could include that separately or using standard LaTeX commands. For example,

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

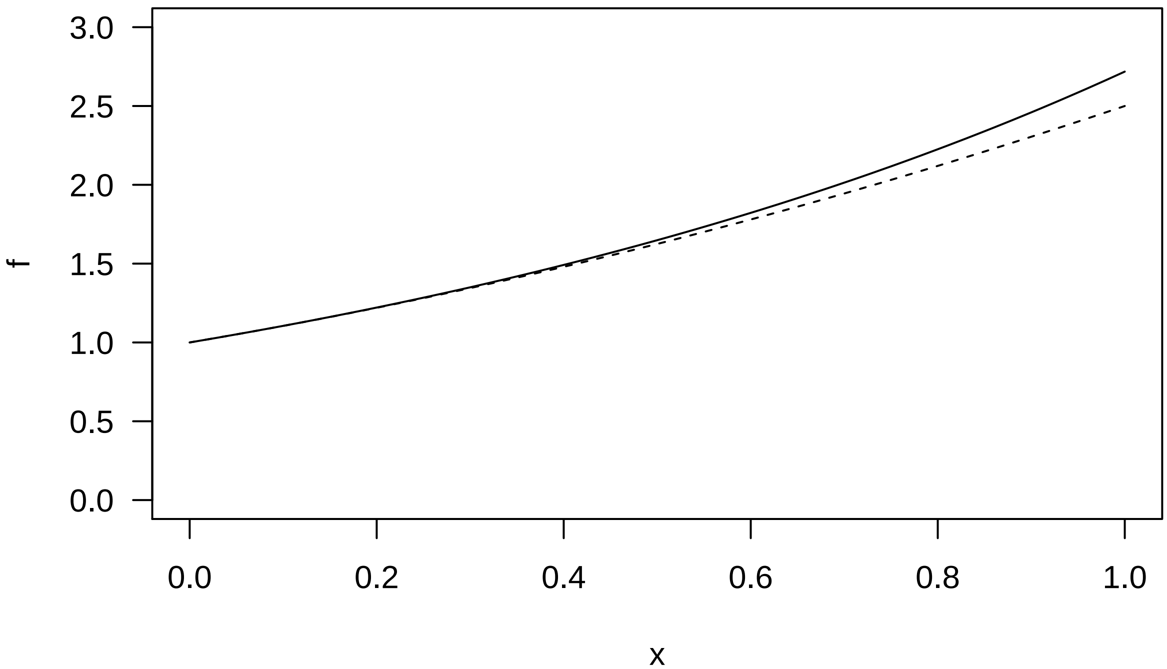
Hermite

A function and derivative.

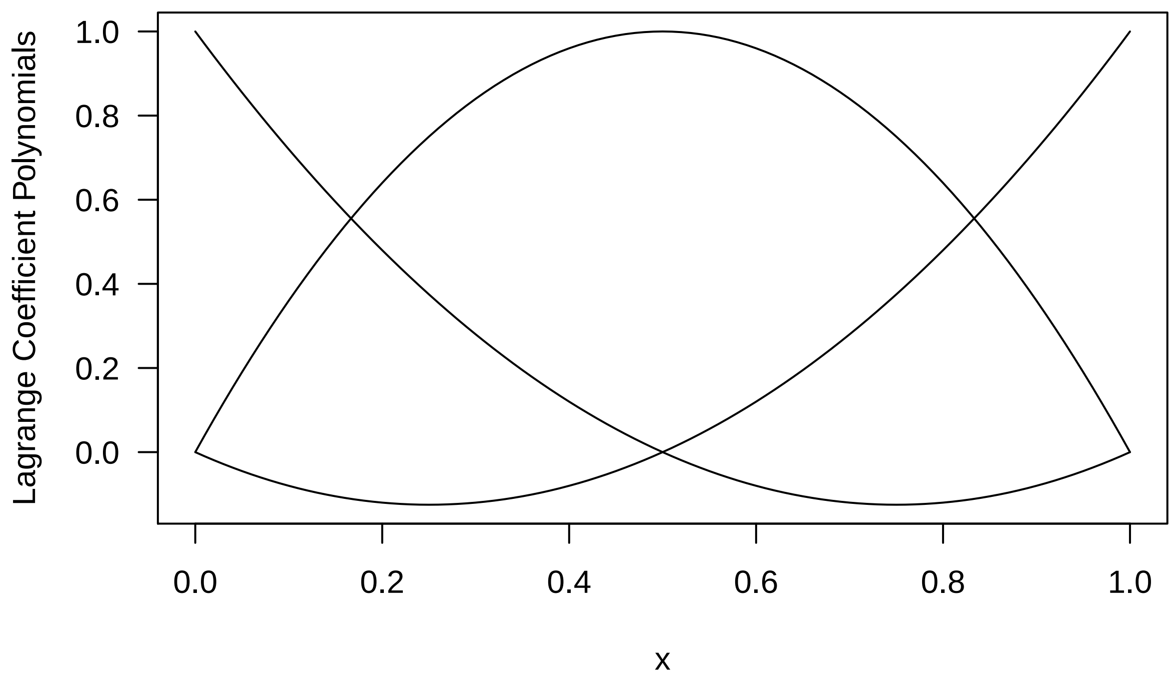
```r
# Original function
f <- function(x)exp(x)
# Could possibly do symbolically or switch to Mathematica
fp <- function(x)exp(x)

# Make a Taylor Polynomial
taylor <- function(x) 1 + x + x^2/2

# List of numbers
xs <- c(0, 0.5, 1)
```

A simple plot.

```r
plot(f, xlim=c(0, 1), ylim=c(0, 3), las=1)
plot(taylor, xlim=c(0, 1), ylim=c(0, 3), lty=2, add=T)
```




```r
# Construction of Lagrange Coefficient Polynomials
L0 <- function(x)((x-xs[2])*(x-xs[3]))/((xs[1] - xs[2])*(xs[1] - xs[3]))
plot(L0, xlim=c(0, 1), ylab="Lagrange Coefficient Polynomials", las=1)
L1 <- function(x)((x-xs[1])*(x-xs[3]))/((xs[2] - xs[1])*(xs[2] - xs[3]))
plot(L1, xlim=c(0, 1), add=T)
L2 <- function(x)((x-xs[1])*(x-xs[2]))/((xs[3] - xs[1])*(xs[3] - xs[2]))
plot(L2, xlim=c(0, 1), add=T)
```




```r
# primes, inelegant
# could do better
L1p <- function(x)(1)/((xs[2] - xs[1])*(xs[2] - xs[3]))*((x-xs[1])+(x-xs[3]))
```

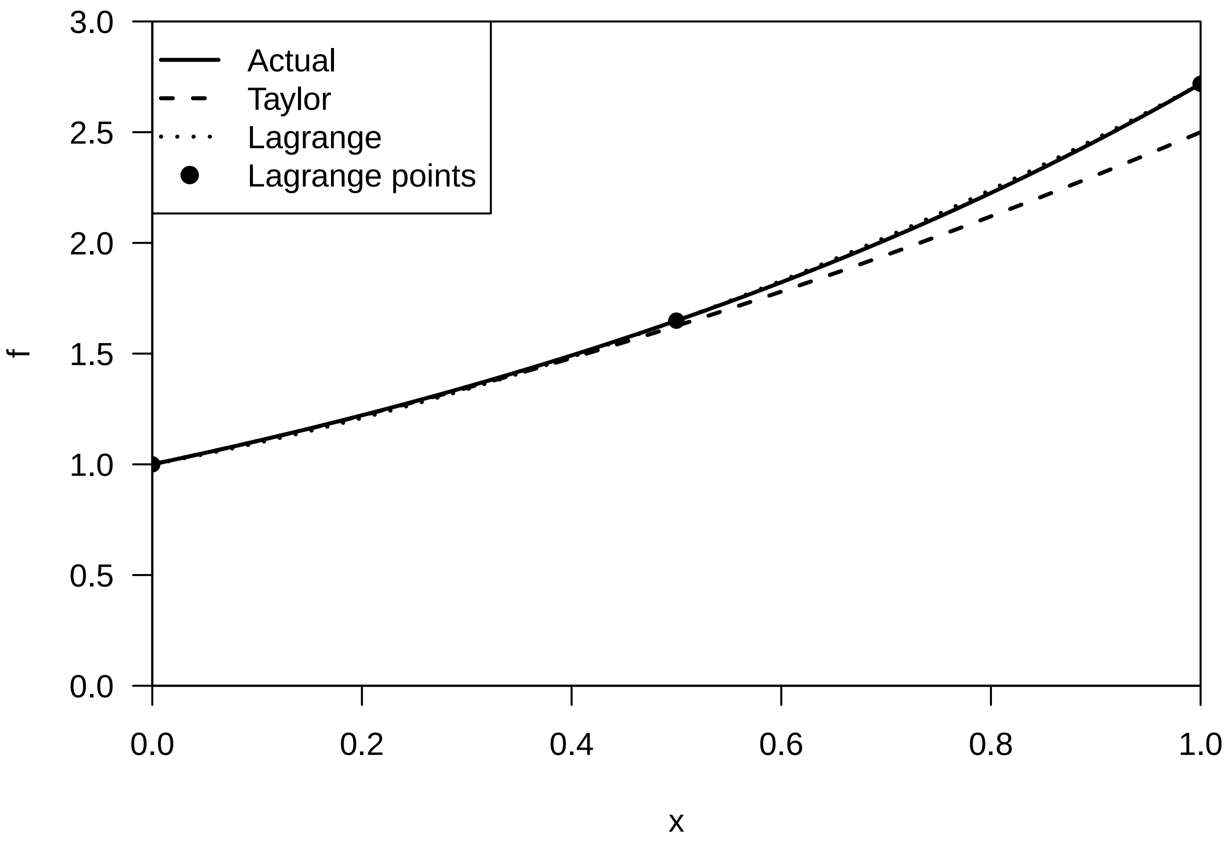
```r
L2p <- function(x)(1)/((xs[3] - xs[1])*(xs[3] - xs[2]))*((x-xs[1])+(x-xs[2]))
L0p <- function(x)(1)/((xs[1] - xs[2])*(xs[1] - xs[3]))*((x-xs[2]) + (x-xs[3]))
```

The Lagrange polynomial

```r
P <- function(x) f(xs[0+1])*L0(x) + f(xs[1+1])*L1(x) + f(xs[2+1])*L2(x)
```

All of them

```r
par(xaxs='i', yaxs='i', mar=c(4.1, 4.1, 1.1, 1.1))
plot(f, xlim=c(0, 1), ylim=c(0, 3), las=1, lwd=2)
plot(taylor, xlim=c(0, 1), lty=2, lwd=2, add=T)
plot(P, xlim=c(0, 1), lty=3, lwd=2, add=T)
points(xs, sapply(xs, f), pch=19)
legend("topleft", c("Actual", "Taylor", "Lagrange", "Lagrange points"),
       lty=c(1, 2, 3, NA), pch=c(NA, NA, NA, 19), lwd=2)
```

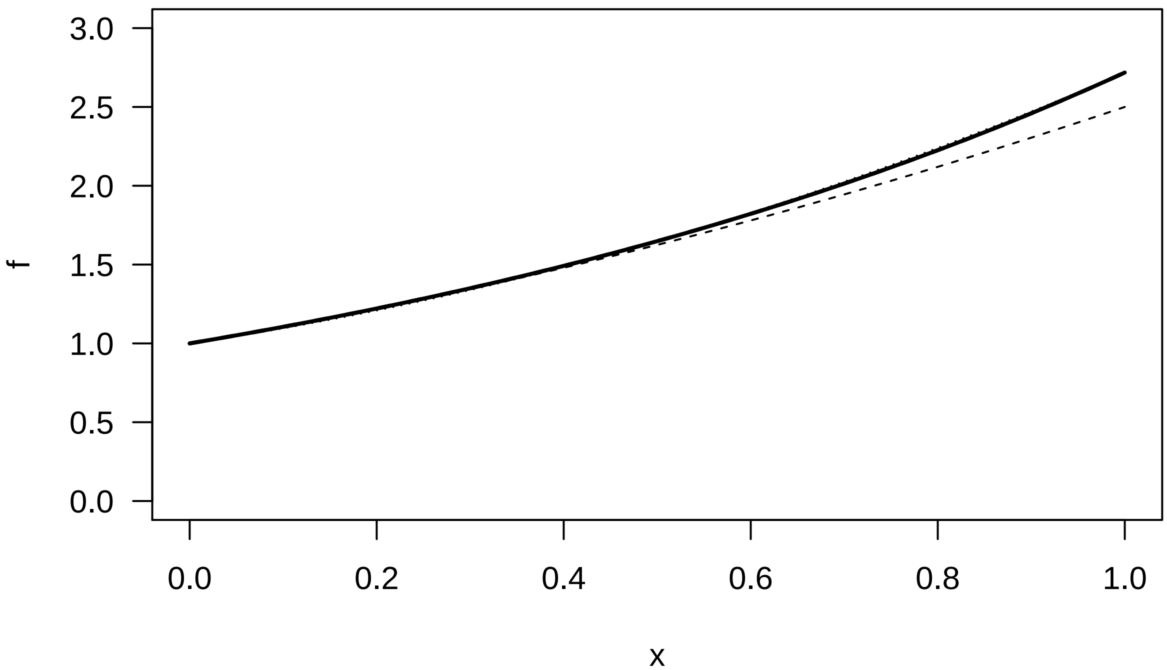


new

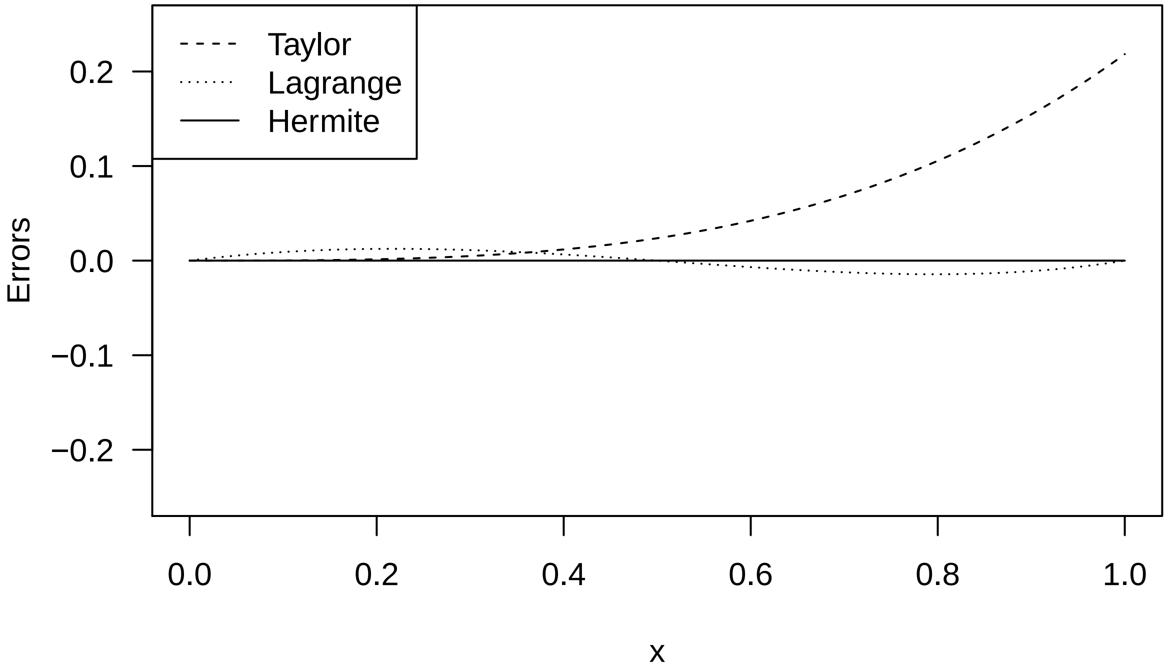
```r
h0 <- function(x) (1-2*(x - xs[[1]])*L0p(xs[[1]]))*(L0(x))^2
h1 <- function(x) (1-2*(x - xs[[2]])*L1p(xs[[2]]))*(L1(x))^2
h2 <- function(x) (1-2*(x - xs[[3]])*L2p(xs[[3]]))*(L2(x))^2

hh0 <- function(x) (x - xs[[1]])*(L0(x))^2
hh1 <- function(x) (x - xs[[2]])*(L1(x))^2
hh2 <- function(x) (x - xs[[3]])*(L2(x))^2
```

```r
H <- function(x) f(xs[[1]])*h0(x) + f(xs[[2]])*h1(x) +
  f(xs[[3]])*h2(x) + fp(xs[[1]])*hh0(x) + fp(xs[[2]])*hh1(x)+ fp(xs[[3]])*hh2(x)
```

```r
plot(f, xlim=c(0, 1),  ylim=c(0, 3), las=1, lwd=2)
plot(taylor, xlim=c(0, 1), lty=2, lwd=1, add=T)
plot(P, xlim=c(0, 1), lty=3, lwd=1, add=T)
plot(H, xlim=c(0, 1), lty=2, lwd=2, add=T)
```




```r
plot(function(x) f(x) - taylor(x), lty=2, xlim=c(0, 1),
     ylim=c(-0.25, 0.25), las=1, ylab="Errors")
plot(function(x) f(x) - P(x), lty=3, xlim=c(0, 1), add=T)
plot(function(x) f(x) - H(x), lty=1, lwd=1, xlim=c(0, 1), add=T)
legend("topleft", c("Taylor", "Lagrange", "Hermite"),
       lty=c(2, 3, 1), lwd=1)
```


Taylor
Lagrange
Hermite
0.2
0.1
0.0
−0.1
−0.2
Errors
0.0
0.2
0.4
0.6
0.8
1.0
x